# Step 7 – Static, final and inheritance

Another basic concept of object oriented programming: Inheritance.

Goals for this step:

- I can create a class that inherits from another class.
- I know how to override a method.
- I understand what static means for fields and methods.
- I understand what the final entails.
- I know the Object class and some of the methods that are there.

Relevant chapters in the book:

- Cap 10: Static methods and variables
- Cap 14: Overriding methods

Here is the questionnaire for this step.

## Task 1

You have to make 4 classes. You have to find out for yourself how the classes relate to each other when it comes to inheritance. Here are the classes:

**Shape**:

- Fields:
  - *color* – What color should the shape have. Datatype: java.awt.Color.
  - *filled* - If the shape is filled with color or not.
- Two Constructors
  - Empty (no parameters). This should set the figure to colored with red color.
  - One that takes both color and filled and sets fields with the values.
- Getters and setters
- A *toString* method that returns a String describing the object.

**Circle**:

- Field: *radius*. Let's use the primitive data type double, then we get some experience with it.
- Three constructors:
  - Tom. Sets the radius to 1.0. Inherits default values from Shape.
  - One that takes the radius. Inherits default values from Shape.
  - One that takes color, filled and radius.En *toString*-metode som returnerer en tekstlig beskrivelse av objektet.
- A *getArea* method that returns the area of the circle.
- A *getPerimeter* method that returns the perimeter.
- Getters and setters

**Rectangle** :

- Fields:
  - *width* – (double)
  - *length* – (double)

- 3 constructors
    - Empty. Sets width to 1.0 and length to 2.0 and filled with green color.
    - One that takes width and length. Sets filled with green color.
    - One that takes width, length, color and filled
- Getters and setters
- A *getArea* method that returns the area of the figure.
- A *getPerimeter* method that returns the perimeter.
- A *toString* method that returns the textual description of the object.

**Square** (– a type of rectangle that has the width equal to the length)

- Field: *side* – (double)
- 3 Constructors
    - Empty.The side is 1.0.
    - One with 1 parameter: *side*
    - One which takes *side*, *color* and *filled* as parameters.
- Let's say that since Square is a type of Rectangle, Square should also have default values "filled with green" if the values are not specified when creating the object.
- Getters and setters
- A *getArea* method that returns the area of the figure. (if needed)
- A *getPerimeter* method that returns the perimeter. (if needed)
- A *toString* method that returns the textual description of the object.

## Task 2

Create a program that tests whether the classes you have created work satisfactorily. Be sure to create the different objects with different constructors and call on methods like toString, getArea and getPerimeter to see if you get the expected result.

Part of the testing involves checking whether the correct default values are set when they are used when creating an object. Here's a little summary:

- Default values for color and filled for geometric shapes are red and that it is filled.
- The default value for the radius of a circle is 1.0.
- Default values for rectangles are filled with green and length = 2.0 and width = 1.0.
- The default value for the sides of a square is 1.0.

## Task 3

Do the classes you have created seem to work satisfactorily? So good! But why not check if Per's tests (and Bogdan's tests that are now up) show the same? Use the test classes located in the github repository and see if your classes work as the tests expect. Make any necessary improvements.

You use the test classes in the same way as we did in Step 2, so take a look there if you need information on using the test classes.

The tests assume that your getter methods have certain names (isFilled, getLength, getArea, etc.). These are based on IntelliJ's autogeneration of getters and setters. So if you have used auto-generation, then it will work fine (as long as you have the same name on the fields as described in the problem).

## Extra Task 1

Create a new class for an optional geometric shape. Create your own rules for default values.

## Extra Task 2

Look at the automated tests from Canvas. Try to create similar tests for your new class in Extra assignment 1. We have not gone through this, but I think there should be sufficient information to rely on in the existing tests.

## Extra Task 3

Do you have even more time to spare? Wow, so good! We have talked about opportunities to find more tasks online. Mass training is gold if you want to be good at programming. I know several of you are already have already mentioned code wars. Why not try that?

Another potential idea: Kattis. Now that we have learned to read from a file, there are many tasks there that you have the opportunity to solve. Such as:

- https://open.kattis.com/problems/everywhere
- https://open.kattis.com/problems/aaah
- https://open.kattis.com/problems/abc

Last I checked, Sweden was ahead of Norway… just saying…

The tasks show the degree of difficulty, so you can find tasks that you think are right for you. Did you find a nice task?